

UNIVERSITY OF AVEIRO

DEPARTMENT OF ELECTRONIC, TELECOMMUNICATIONS AND
INFORMATION

INTEGRATED MASTER IN COMPUTER AND TELEMATICS ENGINEERING

Software Engineering

**M3 SPEC- Full Specification and
Requirements**

2015/2016

Group: g04

João Almeida, 50640

Ricardo Silva, 50427

Hugo Silva, 49900

Index

[Project description](#)
[Project Requirements](#)
[Project Modules](#)
[Module Details](#)
[Features](#)
[Use Cases](#)
[Handicaps](#)

Project description

The goal of this project is to develop a standalone system for reading Portuguese Citizen SmartCards, logging the events (card inserted/removed) and info (card number/name of owner/...) both locally and in an external server (through a message broker), allowing the visualization of the card data and logging history in a locally provided web page and establishing an authentication method which connects each card to a user defined password.

This system is to be implemented as part of the “DETI - Events and rooms manager” project and will provide information to the “Entry Control” module relative to the card events registered.

Project Requirements

Hardware Requirements:

- SmartCard Reader (model used during development:
http://www.bit4id.com/en/index.php?option=com_content&view=article&id=86&Itemid=503&lang=en)

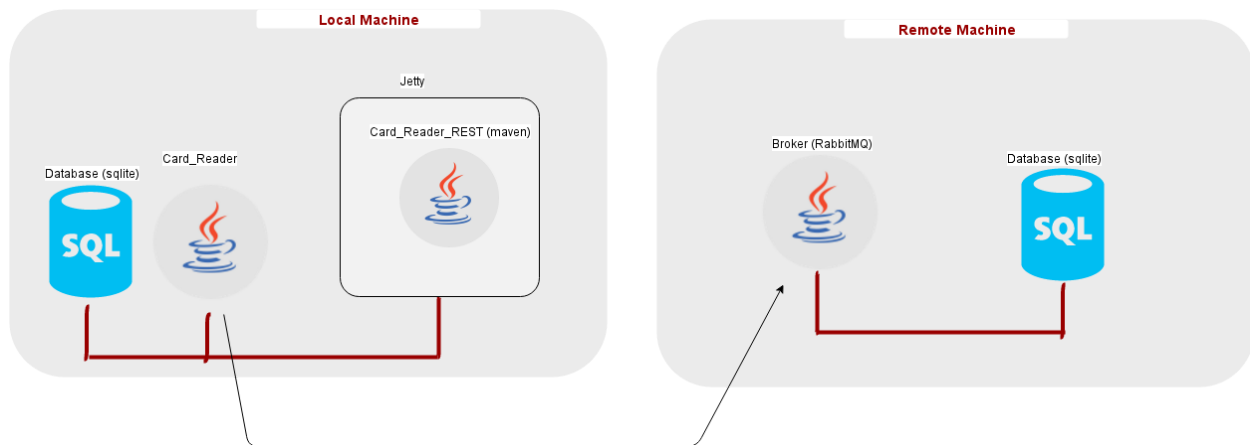
Software Requirements:

- Linux OS (not really necessary but the system is not yet optimized for Windows as well)
- Java (Oracle) JDK1.8 (<http://java.com/en/download/manual.jsp>)
- Portuguese citizen card middleware and certificates
(https://www.cartaodecidadao.pt/index.php?option=com_content&task=view&id=102&Itemid=44&lang=pt.html)
- RabbitMQ service installed and started/enabled
(<https://www.rabbitmq.com/download.html>)
- Sqlite3 (<http://mislav.net/rails/install-sqlite3/>)
- **Jetty Servlet Engine** (<http://download.eclipse.org/jetty/>)

(All java library dependencies are already included along with the Jar and War packages provided.)

Project Modules

This project is composed by 4 different modules (which are described in this section):



- Database

A sqlite Database which stores the list of people that used the Card Reader and their interactions.

This database is composed of 3 tables: person, interaction and current card.

Next are described the table contents:

- Person
 - BI number
 - Country
 - First name
 - Last name
 - Notes
 - Document Type
 - Card Version
 - First name of father
 - Last name of father
 - First name of mother
 - Last name of mother

- SNS number
- Locale
- Delivery date
- Height
- SS number
- Sex
- PAN card number
- Birth date
- Mrz1
- Mrz2
- Mrz3
- Nationality
- NIF number
- Card number
- Delivery entity
- Authentication
- Interaction
 - Type of interaction (card inserted/removed)
 - Room code (i.e.: 4.2.11)
 - Time (in unix time)
 - BI number of card involved in the interaction
- Current card
 - Only holds the BI number of a card currently inserted in the reader)

For a detailed diagram of the database, refer to the annexed image “ES_database.png”.

- Card Reader

Java application which listens for events relating to the smartCard reader (inserted/removed), logs them in the local database and sends them to the broker, in the form of a message containing the following topics:

- Internal id associated with the card
- Room Code
- timeStamp
- Type of interaction (inserted/removed)

In the annexed document “Card_Reader_desc.pdf” there is an interaction diagram detailing the methods and libraries used in the application.

- Local Web Page

Maven application deployed by the Jetty Servlet Engine the provides a local interface (in the internet browser) for the owner of the citizen card currently inserted in the reader to consult his/her information as well as the local logging history associated with the card (a more detailed view of the interface is present in the “Module Details” section).

Jetty Server specifications:

Port: 3389

User Page Path: “localhost:3389/CC_Reader_REST/UserPage_servlet”

- Imaginary Broker

This imaginary broker was used to simulate the interaction the system will have with the “Entry Control” module in a posterior development of the “DETI - Events and rooms manager” project .

So far, the broker is connected to a small database which only has one table containing a list of the interaction associated with the reader.

The message body and database table contain the following information:

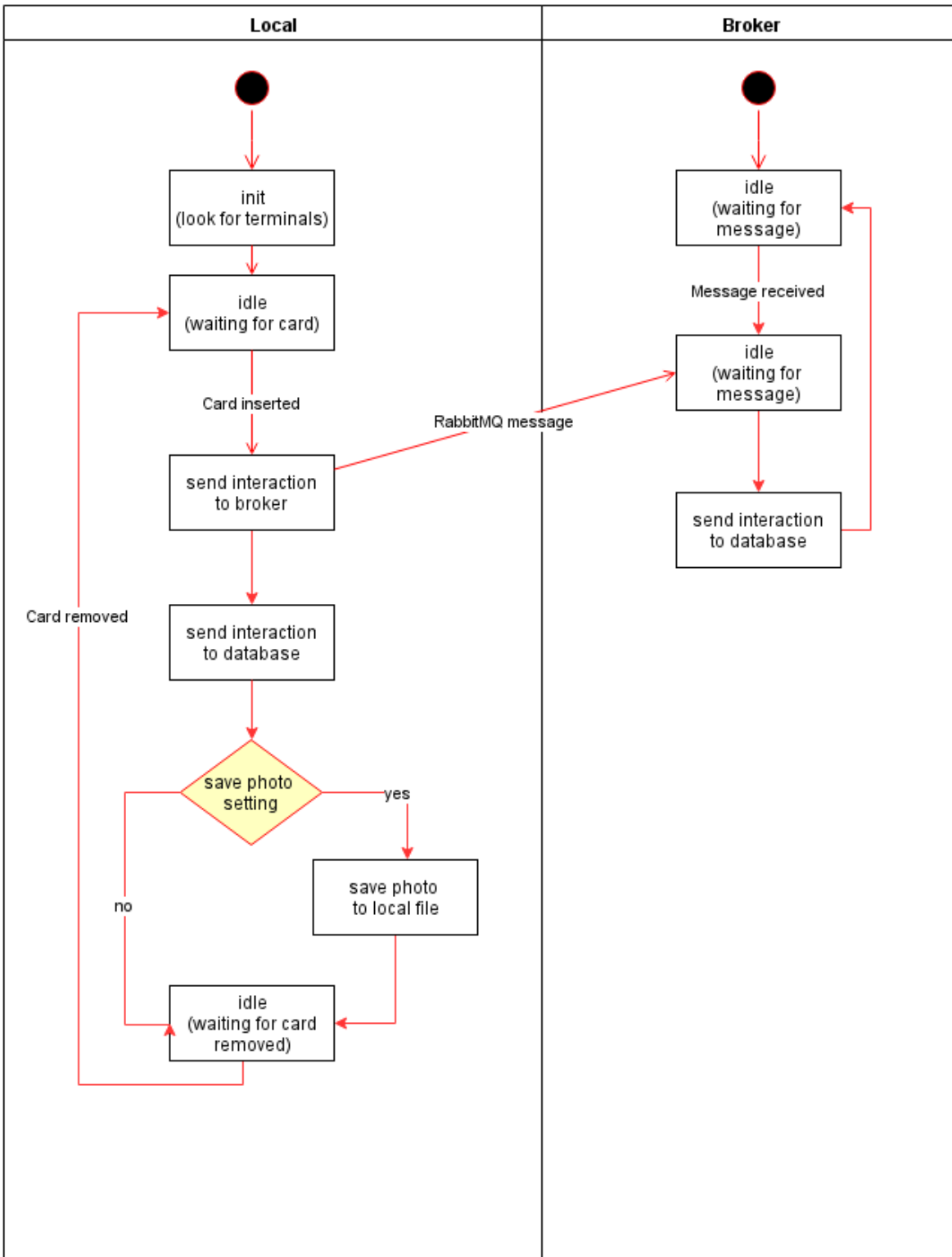
- Interaction (inserted/removed)
- Room Code
- Time (in Unix time)
- Internal ID

At this point, the broker used is a RabbitMQ server deployed by a java application and the message sent consists of a json string.

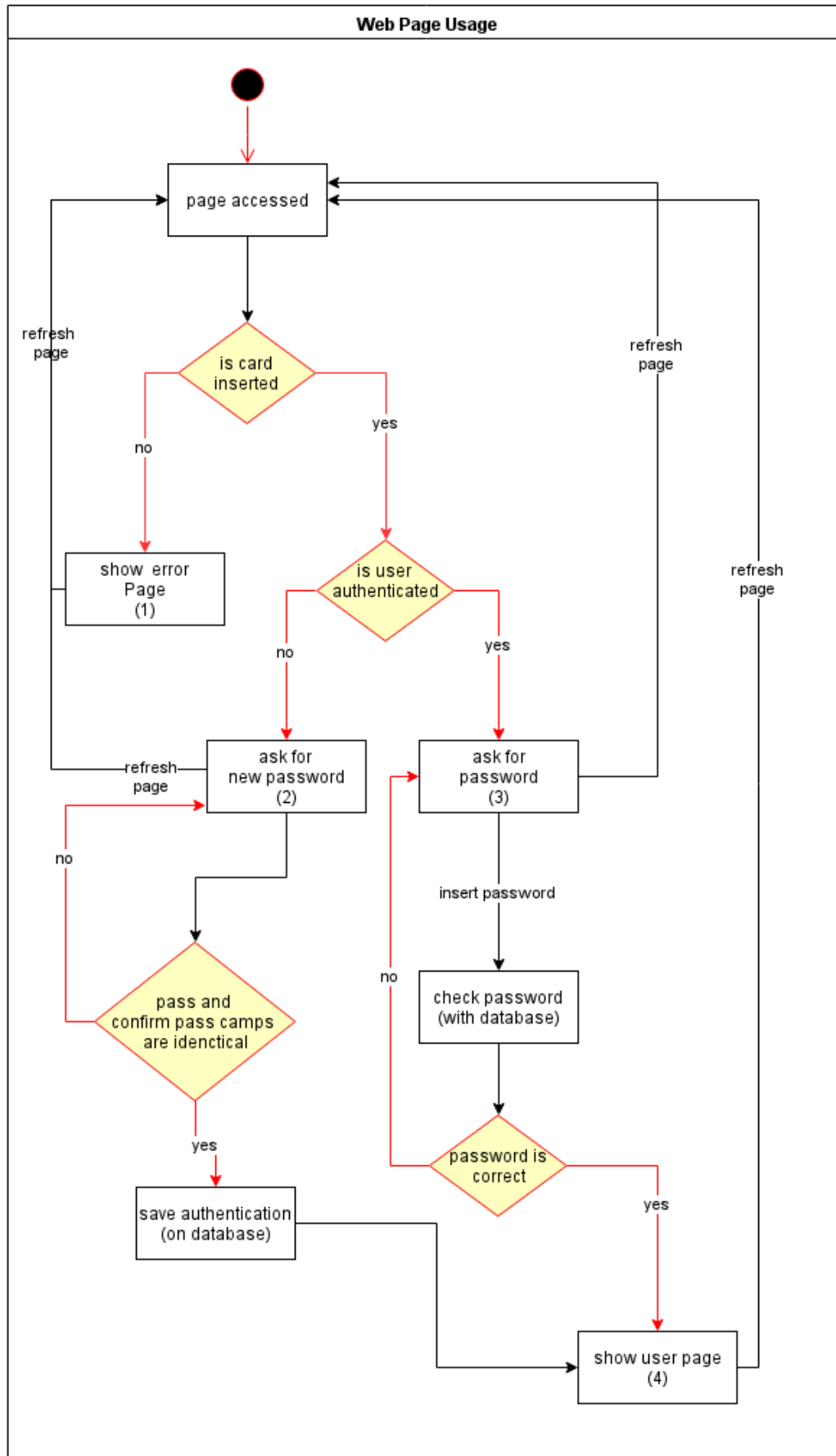
The annexed image “rabbit_msg_json.png” represents an example of the json string used.

Module Details

- Card Reader Cycle (activity diagram):



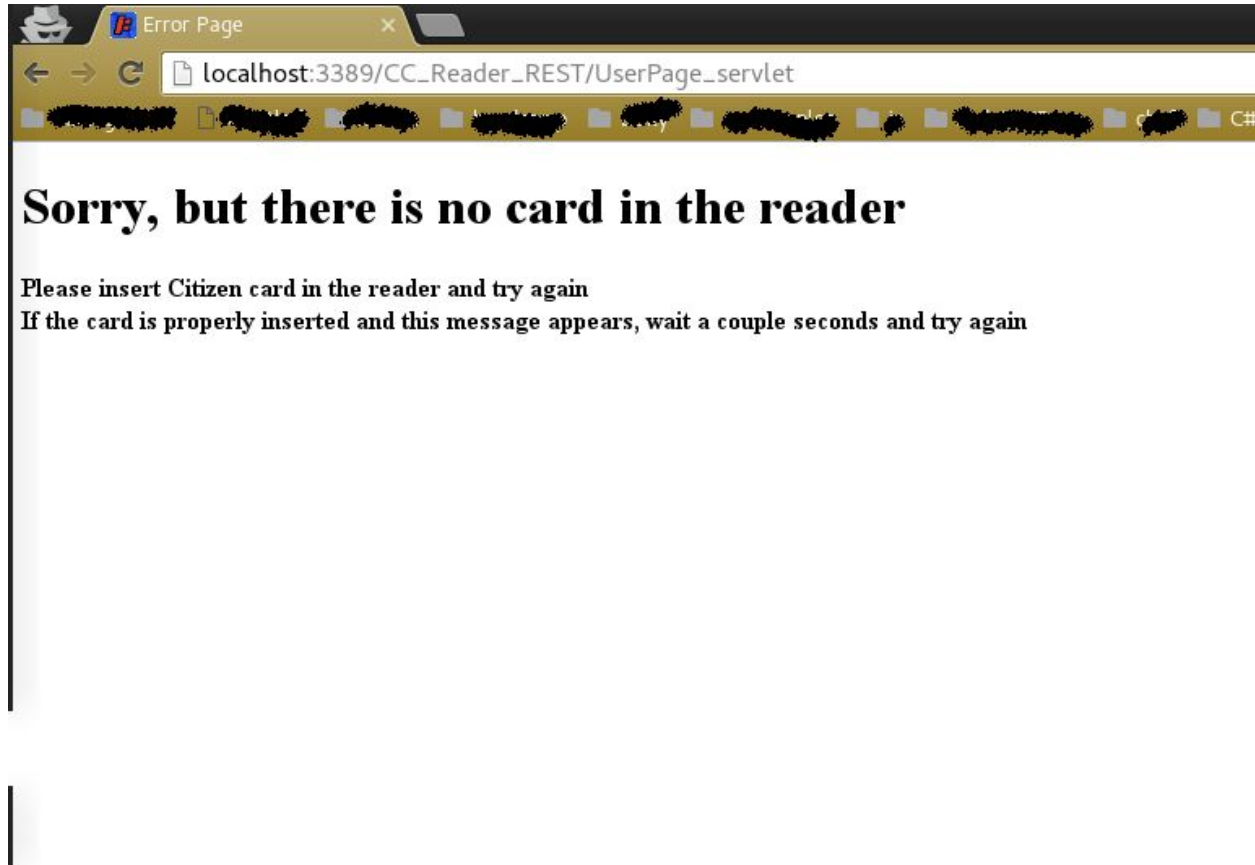
- Web Page usage (activity diagram):



Local Server Pages:

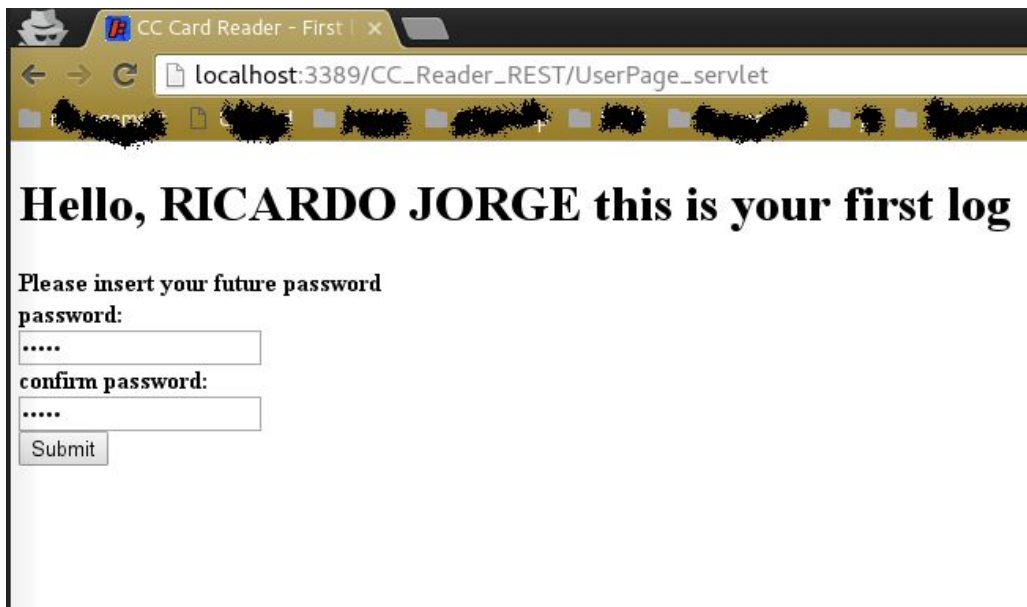
(1) Error Page

This page is displayed when there is no card in the reader or the card hasn't been processed yet.



(2) First Login Page

If the card is inserted but the user still hasn't authenticated his/her account, the server prompts the user to define the password which will be used in the future to access the personal data.



The screenshot shows a web browser window with the title "CC Card Reader - First". The address bar displays "localhost:3389/CC_Reader_REST/UserPage_servlet". The page content includes a greeting "Hello, RICARDO JORGE this is your first log" and a prompt "Please insert your future password". Below this, there are two input fields: "password:" and "confirm password:", both containing five asterisks. A "Submit" button is located at the bottom of the form.

(3) Nomal Login

If the card is inserted and the user already has a password associated to his/her account, the system requires that password in order to move on to the User Data page.



The screenshot shows a web browser window with the title "CC Card Reader - log". The address bar displays "localhost:3389/CC_Reader_REST/UserPage_servlet". The page content includes a greeting "Hello, RICARDO JORGE please log in" and a prompt "password:". Below this, there is a single input field and a "Submit" button.

(4) User Page

After the credentials have been given, the system displays the user's personal data as well as his/her logging data for card related events in the local machine.



[log out](#)

Welcome RICARDO JORGE

Your Data:

numBI: [REDACTED]
BirthDate: 10 [REDACTED]
CardNumber: 139 [REDACTED] 8
CardNumberPAN: 00000 [REDACTED] 556
CardVersion: 004.004.21
Country: PRT
DeliveryDate: 12 12 2014
DeliveryEntity: República Portuguesa
DocumentType: Cartão de Cidadão
Firstname: RICARDO JORGE
Lastname: FREITAS SILVA
FirstnameFather: [REDACTED]
LastnameFather: A [REDACTED] SILVA
FirstnameMother: A [REDACTED]
LastnameMother: PE [REDACTED] SILVA
Height: 1,72
Locale: GICiv. - LC AVEIRO
Mrz1: I
Mrz2: [REDACTED] <<<<<<<<<8
Mrz3: [REDACTED]
Nationality: PRT
Notes:
NumNIF: 2 [REDACTED] 8
NumSNS: 28 [REDACTED]
NumSS: 2 [REDACTED]
Sex: M

Your Logged Interactions:

Room code: 4.2.11	TimeStamp: 2016-04-10 23:27:10 GMT	Action: inserted
-------------------	------------------------------------	------------------

Features

- User data and event logging (both locally and remotely through messaging)
- User account authentication through user defined password
- This system can be built and deployed in any computer (since all the requirements are easy to install)
- System building and deploying executable in one command using bash script (in Linux) (script names: build_and_deploy.sh / deploy.sh).

Use Cases

The use cases defined are only relative to SmartCard owner interactions.

ID: UC-01

Title: register personal account on database

Description: add user data to database and log first interaction (card inserted)

Preconditions: system running, citizen card not expired

Postconditions: user becomes registered in database and can authenticate account

Main Steps:

- 1 - User inserts card in reader
- 2 - waits a few seconds (while application processes card information)
- 3 - user is registered. (optional: if user page is accessed in the browser and “first login” information is displayed, this serves as confirmation of successful registering.)

Possible errors:

- 2 - card is removed unexpectedly or is corrupted -> register action is canceled.

Frequency of use: frequent at start but decreasing with time.

Status: working.

ID: UC-02

Title: authenticate account on database

Description: establish password associated with user account on database

Preconditions: system running, citizen card inserted in reader, user registered(UC-01)

Postconditions: user authenticates account and becomes able to enter User Page

Main Steps:

- 1 - User goes to User Page URL in browser
(localhost:3389/CC_Reader_REST/UserPage_servlet)
- 2 - “first login” page is displayed
- 3 - user fills “password” and “confirm password” camps and submits.
- 4 - user’s account is now authenticated and User Page is displayed

Possible errors:

3 - “password” and “confirm password” camps don’t match, the system asks for the credentials to be inserted again.

Frequency of use: frequent at start but decreasing with time.

Status: working.

ID: UC-03

Title: Check personal information

Description: consult personal card data and logging history.

Preconditions: system running, citizen card inserted in reader, user registered and authenticated(UC-02)

Postconditions: none.

Main Steps:

- 1** - User goes to User Page URL in browser
(localhost:3389/CC_Reader_REST/UserPage_servlet)
- 2** - “login” page is displayed
- 3** - user fills “password” camp and submits.
- 4** - User Page is displayed

Possible errors:

3 - “password” camp doesn’t match the password stored in database, the system asks for the credentials to be inserted again.

Frequency of use: frequent.

Status: working.

Handicaps

In this section we will list the known bugs affecting the system and some of the betterments to be developed in posterior iterations.

Bugs:

- Wrong type of card inserted (gets caught in try catch and doesn't break program, but system stops being able to read citizen cards properly after this occurs)

Modules Under Development:

- Authentication remains local (because imaginary broker is only receiving messages and not replying, a user can have different passwords for different machines)

Possible solutions

- Responsive broker that, when receiving event checks for changes in the user's password and returns it.
- Local broker (using a local broker in each machine that listens for "password changed" messages from remote server)

The final solution must be discussed with other groups.

Priority: High

- Security issues (database is accessible by anyone using the computer in which the system is running)

Possible Solutions:

- Ciphred database (using hardcoded password)

Priority: High

- Ugly interface (the web page interface can be better)

Priority: Low

- Password changing (so far, the password defined by the user in the first login is final, but it should be redefinable at any time)

Priority: Medium

- Rest API (for use by the “Entry Control” to retrieve information according to the internal ID associated to the card sent in the messages to the broker)

Priority: High